Towards a Dynamic Policy Enhanced Integrated Security Architecture for SDN Infrastructure

Kallol Krishna Karmakar*, Vijay Varadharajan*, Uday Tupakula*, Michael Hitchens†
*Advanced Cyber Security Engineering Research Centre
The University of Newcastle, Australia -2308
† Department of Computing, Macquarie University, Australia - 2113
[kallolkrishna.karmakar, vijay.varadharajan,uday.tupakula]@newcastle.edu.au*, michael.hitchens@mq.edu.au†

***Abstract*—Enterprise networks are increasingly moving towards Software Defined Networking, which is becoming a major trend in the networking arena. With the increased popularity of SDN, there is a greater need for security measures for protecting the enterprise networks. This paper focuses on the design and implementation of an integrated security architecture for SDN based enterprise networks. The integrated security architecture uses a policy-based approach to coordinate different security mechanisms to detect and counteract a range of security attacks in the SDN. A distinguishing characteristic of the proposed architecture is its ability to deal with dynamic changes in the security attacks as well as changes in trust associated with the network devices in the infrastructure. The adaptability of the proposed architecture to dynamic changes is achieved by having feedback between the various security components/mechanisms in the architecture and managing them using a dynamic policy framework. The paper describes the prototype implementation of the proposed architecture and presents security and performance analysis for different attack scenarios. We believe that the proposed integrated security architecture provides a significant step towards achieving a secure SDN for enterprises.**

***Index Terms*—Enterprise Network Security, SDN Security, Policy-based Security Architecture, Network Attacks.**

I. Introduction

SDN is changing the texture of modern networking, making it more flexible and programmable, moving away from the current control protocols dominant in the TCP/IP Internet stack. It has the potential to change the way networking is conducted by enabling devices that are open and controllable by external software. This is in contrast to today's proprietary network equipment that has fixed protocols embedded into them by the vendors. SDN opens up new avenues of research to realise network capabilities that until now were impossible or extremely cumbersome, thereby helping to make future networks more manageable and practicable. In SDN network architecture, control logic is decoupled from the data plane devices. The driver of the SDN architecture is a centralised set of software modules called Controller, which acts as a network operating system. Like any other operating system, it provides a platform to develop applications. These applications drive the data plane devices and, hence help control the network. With such sophisticated networking architecture, SDN can achieve tremendous networking flexibility. For instance, network applications can dynamically restore a compromised network device or reroute network flows based on the dynamic context of the network flow. Hence, it can help to improve the security and performance of the network domain. Thus, more and more enterprises are embracing SDN [1].

Enterprise networks typically consist of a large number of users, resources, devices and complex network topology, several different security measures and policies. Also, the pattern and frequency of the data communication and sharing through these devices among users and devices make the enterprise networks more complex [2]. Moreover, the enterprise networks have their own security issues [3]. For instance, adversaries with BYOD devices may be able to easily penetrate the enterprise network perimeters. This is often due to the lack of coordination between different and somewhat separate enterprise security mechanisms and policies and configurations causing loopholes in the complex network setup. Furthermore, to enhance the network services, enterprises sometimes deliberately disable monitoring of some devices, for instance, network printers and remotely accessible IoT devices. However, for an adversary, these provide the easiest options to penetrate the network system. For instance, the attack in Jet Propulsion Laboratory of NASA in April 2018 used a Raspberry Pi unit to create a backdoor and steal valuable information [4]. Also, the conflicts between different enterprise network policies can lead to security issues. Furthermore, SDN is still in its infancy in terms of its development and deployment, which can lead to security vulnerabilities in their architectures [5], [6], [7].

Deployment of SDN in an enterprise domain requires the design of a holistic security architecture and a careful consideration of various design choices. In this paper, we present an Integrated Security Architecture for SDN, which we believe provides a significant step towards achieving a secure SDN for enterprises.

An important characteristic of the Integrated Security Architecture is its ability to deal with dynamic situations. Hence a core element of our architecture is a dynamic policy-based framework that can respond to changes. In the context of security such changes can be changes in the security attacks or the trust associated with the various components or services in the infrastructure. Consider the situation where the security architecture consists of various security services and mechanisms to detect and counteract different types of attacks in the SDN. For instance, it may have attack detection and network flow verification components as well as security mechanisms to communicate with the SDN Controller. In an enterprise

978-1-7281-4973-8/20/$31.00

network, there is a need for dynamic coordination of these security services and their operations, which is provided by our policy-based integrated security architecture. The dynamic policy framework achieves coordination of different security mechanisms from detection and mitigation to enforcement. For instance, we have policy-based detection attacks; then following attack detection, the policy framework can dynamically create mitigation policies and enforce them at the data plane. Let us assume that in an enterprise SDN network, a malicious user creates a spoofing attack using the IP/MAC address of a legitimate host and launches attacks against the SDN Controller (as well as the switch connected to the host). First, the security mechanisms in our architecture will detect the malicious host and the connecting switching device. Then the dynamic policy framework will create mitigation security policies. In this case, the mitigation policy will consist of the design of flow rules to counteract the attack, which can involve dropping specific flows (or even re-directing them to certain quarantined locations for analysis). Then these mitigation policies will be enforced at locations nearest to the originating attacking device/host, as well as helping to reduce the resources (e.g the bandwidth) affected by the attack.
Another situation might arise due to change in the levels of trust associated with the network devices. For instance, assume a switch which was previously trusted by the SDN Controller has been attacked and has now become malicious. When this happens, the dynamic policy framework can change the flows in such a way that the malicious switch is not being used in the provision of end-to-end services. Hence an important characteristic of the proposed integrated architecture is the adaptability of the architecture to dynamic changes, which is achieved by having feedback between the various security components/mechanisms in the architecture and managed by a dynamic policy framework. The contributions of the paper can be summarised as follows:

- We present an Integrated Security Architecture (ISA) for SDN enterprise network.
- The proposed ISA uses dynamic policies for securing the enterprise network infrastructure. There are two types of dynamic policies. One that deals with counteracting attacks and the other helps with the specification and routing of secure flows across the SDN network.
- We have developed a prototype of the proposed security architecture using a VMware NSX cluster.

The paper is structured as follows. Section II discusses the security requirements and the various components and mechanisms of our ISA security architecture. In Section III, we present a prototype implementation of the ISA. This section also describes the performance and security analysis of the ISA. Relevant related works are discussed in Section IV. We conclude the paper in Section V.

II. ISA

In this section, we will describe our Integrated Security Architecture and its components. This section is divided into three subsections. First, we outline the security requirements for an ISA in an SDN enterprise domain. Second, we present a high-level picture of the ISA. Finally, we describe the various modules of ISA in detail.

A. Security Requirements

The security requirements for an ISA for SDN enterprise domain are as follows.

R1: Both SDN and legacy enterprise networks use several security services and they are often maintained and configured manually. For instance, configuring and maintaining firewalls, routing and mitigation policies for the entire network, security services to block attacks etc. However, manual setup and the lack of coordination or feedback between the various security services can create security loopholes and jeopardize the security of network infrastructure. Hence, there is a need for feedback between security services, for better security reasoning about changes, e.g. due to changes in security attacks.

R2: Often SDN enterprise network consists of heterogeneous devices (such as IoT devices) and services. Such heterogeneous devices are resource-constrained and are prone to security vulnerabilities, which can lead to network compromises. Also, in an enterprise network, isolated network anomalies detected by different security services may not provide a better indication of new network attacks, such as zero-day vulnerabilities. Hence, there is a need for detecting such anomalous behaviours and provide a translation of behaviours into network compromises.

R3: One of the significant problems in a SDN enterprise network is the lack of trust between the OpenFlow switches and the Controller. The switches maintain communication with the Controller via a secure channel using TLS. However, at any point in time, there is no mechanism present in the OpenFlow switch to detect or evaluate the OpenFlow switch status. Hence, the need for security mechanisms which can determine and maintain the trust level of the OpenFlow switches, which in turn can be used to construct secure routing policies.

R4: The SDN enterprise network should have security mechanisms to maintain user flow confidentiality and integrity. These are not new. However, the integration of these security mechanisms is essential.

R5: The Integrated Security Architecture should be able to detect various attacks, not only using the traditional mechanisms but also with the analysis of the coordinated features proposed in the integrated architecture.

B. Integrated Security Architecture (ISA)

Our SDN-based ISA architecture for enterprise network consists of four tiers (shown in Figure 1). They are as follows:

User/End Device Tier: In most cases, end-users/hosts, IoT devices reside in this tier. If required monitoring agents can be included in these devices.

Switching Tier: The switching equipment such as OpenFlow switches, routers, firewalls etc. are present in this layer. All of them are OpenFlow supported switching devices. There can be switching devices running over

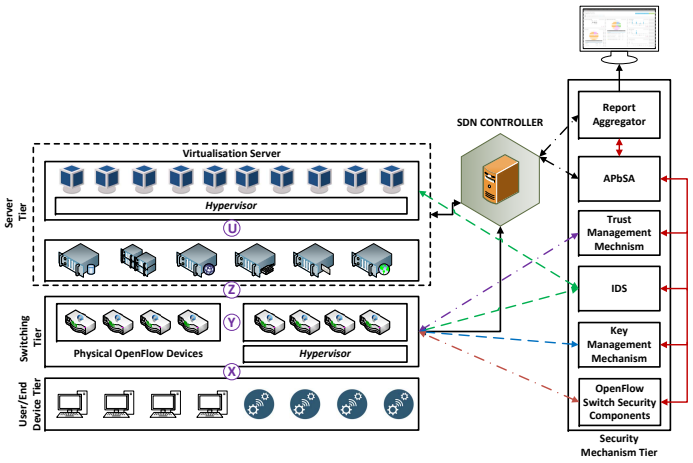


Fig. 1: Integrated Security Architecture using SDN for enterprise network

virtualised switching network, i.e., software routers or Open vSwitches.

Server Tier: In this tier, different types of servers are connected. For instance, web server, application server, content management server, directory server, email server and data management server. It also hosts a virtualisation server which hosts the virtual machines. The hypervisor has virtual machine monitoring capability.

Security Mechanism Tier: The Security Mechanism Tier consists of the security mechanisms used in ISA. They work in coherence to provide security services to the whole enterprise network infrastructure. The major components of this tier are Authorization Policy-based Security Architecture (APbSA), Trust Management Framework, Intrusion Detection System, Key Management Mechanism and OpenFlow Switch Security Components. The architecture also introduces a report aggregator module which collects and stores the reports.

The network connections between the end-users and the switching tier are represented as X. The internal connection between the physical and virtual switching hardware is represented as Y. The internal connection between the physical server and the virtual server is represented as U. The server tier maintains communication with the switching tier via Z.

To describe the functional high-level overview of the Integrated Security Architecture, we will take a bottom-up approach. Here, we will start with the user/end-device tier. The end hosts are normal end-user machines or IoT devices. They have usual processing power, TPM and authentication capabilities. In some cases, it is possible to integrate an extra authentication mechanism for devices. For example, as IoT devices are prone to attacks and vulnerabilities, an extra light-weight authentication mechanism can be added to this architecture. We have presented such, IoT specific architecture in [8]. This architecture uses OAuth and Elliptic Curve Cryptography (ECC) for authenticating IoT devices.

The switching tier consists of modified OpenFlow switches. They can be of two types, virtual and physical and every switch is equipped with OpenFlow Switch Security Component. It has four functional components integrated into the Component. We call each of the functional components as agents. The first agent helps to enforce flow-based policies at a very fine-granular level. The second agent can validate state, user and network flows. The third agent has IDS capability and have a storage facility to cache local rules, signatures, etc. The final and the last agent is responsible for measuring and reporting the status of the OpenFlow switch. The switch is also capable of encrypting different flows and switch communications.

The servers can be virtual or physical; they are all monitored. They have IDS capabilities. The driver of this whole Integrated Security Architecture is Authorization Policy-based Security Architecture (APbSA). Firstly, it uses a fine-grained policy language. The policy language captures various features of the network flow, allowing the architecture to establish a path and flow-based policies. APbSA policies restrict both physical and virtual network communication. Secondly, it coordinates all the security mechanisms present in ISA. Finally, it hosts a report aggregator or log analyser. The aggregator module looks for system and network event causalities to identify potentially malicious events. With the help of APbSA, it can alert the system admin. The report aggregator also presents all the SDN enterprise network activities in a dashboard.

Now, we will explain a high-level functional overview of the Integrated Security Architecture with an example. Let us assume that the network administrator has already installed some network policies in the APbSA. Also, the IDS module has an updated set of Snort signatures. Let us consider a network attack for which the IDS does not have any installed signature and the APbSA does not have any flow blocking/-dropping policy for the device/user launching the attack. In this scenario, the OpenFlow switches residing in the switching tier are equipped with a Logic Bomb malware. A logic bomb malware is a type of malware that executes itself after a certain logic or sequence of events are fulfilled. In this case, the logical trigger was a special packet from a specific IP. The OpenFlow switch passed the boot-time TPM assessment, and it was clean as expected. However, the adversary triggers the malware by sending an external request from the specific IP address with that special packet payload. Thus, malware becomes active. The target of the malware was to launch a DDoS attack towards a specific host. In this case, APbSA flow authorisation policies will permit as there will be no policies for this particular flow. Also, IDS module will be unable to judge the attack, as it does not have the specific signature. In our, security architecture, trust management mechanism continuously evaluates the status of OpenFlow switches and sends a status report called Switch Status Report (SSR) to the APbSA module. Also, neighbouring switches can send behaviour recommendations. As the infected switch starts to flood, the neighbouring switch recommendations fall, and also the SSR report shows malicious behaviour, resulting in flagging the switch as untrusted. The APbSA dynamically installs drop rules to all neighbouring switches to isolate the infected switches communication. In some cases, it can reset the infected OpenFlow switch too by flushing the flow rules

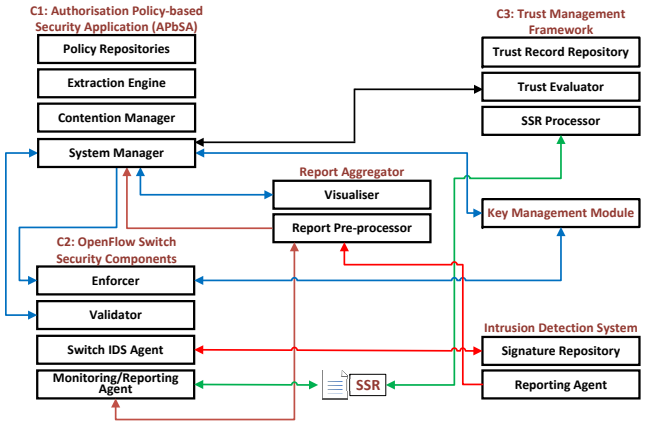


Fig. 2: Components of ISA in Security Tier

or by asking it to restore the switch OS. If both of these approaches fail, this will require human intervention.

The ISA is structured upon the components present in the security tier. Here, we will describe each component, their purpose and how they function in details. Figure 2 shows the component layout and their communication map.

C. C1: Authorization Policy-based Security Architecture:

APbSA is authorisation policy framework for SDN domains. It uses a granular policy language to create and enforce flow and path specific policies for SDN communication. The notion of granularity comes from the amount of the constraints it can accommodate in a policy expression. The APbSA policies can put constraints over users, flows, services, IP, MAC, port, etc. Thus, allowing it to control and manage the flow communication in the OpenFlow switching tier. The constraints are created from different attributes of the packet flow. The attributes are: **(a) Flow Attributes:** Flow ID, sequence of the Flow, type; **(b) Device Attributes:** ID of specific devices; **(c) Switch Attributes:** Switch ID and Security/Trust Label; **(d) Host Attributes:** Source/Destination Host ID; **(e) Flow and Domain Constraints:** Constraints such as Flow Constraints (FlowCons) and Domain Constraints (DomCons) associated with a specific device Flow; **(f) Services:** for which the policies are applicable (e.g. FTP storage access); **(g) Time Validity:** Policy expiration time; and **(h) Path:** Layouts the communication path for any specific user communication.

The APbSA has a modular structure and uses logic to enforce policies. The major components of the APbSA are:

Policy Repository: This is a JSON database which stores the flow- and path-based authorisation policies. Individual policies in the database are termed as Policy Expression (PE). There are two ways to create and store PEs in the repository, i) Creating policies manually (by the administrator.) and ii) the System Manager forms dynamic PEs to control the packet flow in the switching tier. The enforcement of these PEs are coordinated by the System Manager and the enforcer module in the OpenFlow Switch Security Components.

Extraction Engine: The extraction engine firstly helps to extract the flow attributes. It allows the System Manager to run queries in the Policy Repository for matching attributes. Finally, it can send comparison information/report back to the System Manager.

Contention Manager: As the Policy Repository accepts both human and PEs formed by System Manager, there is a huge possibility for policy conflicts. The Contention Manager checks for policy conflicts in the Policy Repository before they are enforced at the switching tier.

System Manager: The System Manager is the command centre of the whole ISA. It manages all the components in the ISA. It provides a command interface to add/update policies and control other components. The tasks performed by the PM are: i) Initialisation, security and trust level assessment for the OpenFlow switches; ii) Creating secure communication profile using Key Management Module; iii) Checking policy conflicts using Contention Manager; iv) Forging dynamic policies to counteract or mitigate threats in the tiers and v) Enforce flow rules in the switching tier.

D. C2: OpenFlow Switch Security Components

In our ISA architecture, the OpenFlow switches are modified to implement various security mechanisms. The security mechanisms deployed in the OpenFlow switches are termed as Switch Security Agents or simply agents. The modified OpenFlow switch empowers ISA to enforce flow and path specific polices. The switch can validate and monitor flows using the agents. The modified switches can also assess switch status and report it to the Trust Management Framework. This report is known as Switch Security Report (SSR). At run-time, the Trust Management Framework can assess the trust level of any OpenFlow switch by using SSR.

The OpenFlow Switch Security Component has four major components. They are as follows;

Enforcer: The Enforcer module is the communication interface between the Controller services (e.g C1: APbSA) and OpenFlow switches. First of all, it helps the System Manager to parse *packet_IN* requests. After that, the System Manager uses Extraction Engine to extract attributes and compare them with the PEs. The most important task performed by the Enforcer agent is to enforce the flow rules sent by the System Manager.

Validator: The Validator agent has two purposes; first it validates the source address and secondly, it validates the flow rules that are installed in the OpenFlow switches.

The Validator agent validates source addresses of any device connected to the source and destination OpenFlow switch. It uses a modified form of Source Address Validation Architecture [9] to validate the addresses. This address validation mechanism helps to stop various type of spoofing attacks and attacks that use spoofing at the initial phase of the attack.

Here, the Validator maintains a database containing the assigned IP, port and specific device MAC addresses. The Validator populates the database whenever a device is connected to the OpenFlow switch port and an IP is assigned. The System Manager in APbSA needs to authorise any changes performed by the Validator. Hence, the likelihood of spoofing attack would be very less.

The System Manager in APbSA coordinates the flow valida-

tion process. Here, the System manager maintains a database of the flows installed for a particular switch. The database keeps a hash of the flows for a particular switch in the database. The System Manager can regularly or randomly request Validator to send flow rule hashes for any specific OpenFlow switch. In the OpenFlow switches, they are stored in normal format. Upon request from the System Manager, the Validator creates a hash of the existing flow rules and send them to the System Manager. A simple comparison of them helps in preventing fake flow rule injection attacks.

Switch IDS Agent: Switch IDS agent uses signature-based attack detection mechanisms to detect the attacks closer to the generating host. It uses python regex for analysing the network packets. The central IDS module(C4) manages and maintains this Switch IDS agent. The attack signatures are cached locally by the Switch IDS agent for faster execution. The central IDS module updates the local signature cache where ever there is any update to the rule database.

Monitoring/Reporting Agent: This is a trusted agent, which is a part of the Trust Management Framework (C3). It helps to measure the status of the OpenFlow switch and sends a status report to the SSR processor module.

At first, this module performs a boot time TPM assessment of the OpenFlow switch. After the TPM assessment, it performs the first status assessment. The agent checks the status for two types of switch events; Internal event and the external event. An internal event is an event that happens inside the OpenFlow switch modules/software/hardware components, i.e., stopping any process, the addition of any process, addition/deletion of any flow rules in Flow/Group tables, opening/blocking any ports, activating/de-activating any hardware interfaces, etc. On the other hand, external events are the events that happen due to external entities connected to the OpenFlow switches. An external entity can be any host machine or an IoT device or other neighbouring switches (OpenFlow or legacy) connected to the OpenFlow switch. It then prepares an initial Switch Status Report (SSR) and transmits it to the SSR processor module. This initial report creates a base level for an OpenFlow switch. After that, the Trust Management Module is programmed to assess the switch trustworthiness in a regular time interval.

E. C3: Trust Management Framework

The Trust Management Framework evaluates the trustworthiness of any OpenFlow switch at run-time. It collects the OpenFlow switch status using Monitoring/Reporting Agent. Our Trust Management Framework then uses subjective logic to convert the status report (SSR) and measure the trust levels [10], [11], [12]. After that, they are stored in the Trust Repository for future evaluation and decision making purposes. The individual component functions are as follows:

SSR Processor: The SSR Processor extracts information from the SSR report. It represents them in a categorised format so that the Trust Evaluator can use the information. For SSR reporting our agent considers only two types of incidents respective to an OpenFlow switch, they are internal and external events. The internal events reflect two type of incidents, i) the number of processes running in the OpenFlow switch at the time of evaluation and ii) the active flow rule at the time of evaluation. The external events are classified into two categories, i) The traffic flow quality, informing the Trust Evaluator about the attack and malicious activities and ii) the recommendation for the neighbouring switches behaviours.

From the definition of the OpenFlow switch internal and external event, we infer two major properties that need to be satisfied for an OpenFlow switch to be trusted by the Controller. They are:

Definition 2.1: **Property 1 (P 1):** Property 1 says that, the OpenFlow switch is not attacked or modified by any malware. Its internal processes are intact and functioning perfectly. Any OpenFlow switch that satisfies this property is considered to be a malware-free OpenFlow switch.

Definition 2.2: **Property 2 (P 2):** According to property 2, an OpenFlow switch is not attacked by any external adversary. The normal external attacks are spoofing, packet injection, header modification, etc. Any OpenFlow switch which satisfies this property is considered to be not affected by any external attacks.

Trust Evaluator: The Trust Evaluator, uses the SSR report to create an evidence table. An evidence table stores trust relationships for future trust evaluation. It contains the trust entities (trustor and trustee), their opinions, experiences (positive, negative and uncertain) with respect to trustee and timestamps of individual experience. In our case, Controller is the trustor and the OpenFlow switches are the trustee. The Controller trusts these OpenFlow switches if and only if it satisfies certain properties, in our case, we have defined two properties, $P1$ (*Def* 2.1) and $P2$ (*Def* 2.2). For this purpose, the Controller uses the SSR report of each OpenFlow switch. Finally, subjective logic is used to calculate overall trust. For an OpenFlow switch, the trust value ranges from 0.00 to 1.00. The Trust Evaluator divides it into four trust levels, T1 (0.00-0.24), T2 (0.25-0.49), T3 (0.50-0.74) and T4 (0.75-1.00). The higher the trust level, the more trusted it is.

Trust repository: The trust repository is a table which holds the trust relationships. It is a complex JSON repository, which stores all the trust relationships for every OpenFlow switch active in the network infrastructure.

F. C4: Intrusion Detection System

This is a simple signature-based Intrusion Detection System, which uses snort signatures for detecting various types of network attacks. The Signature Repository stores these signatures. It also runs queries to update the signature repository regularly. The ISA can have IDS agents in different places, namely, in OpenFlow switches, in hypervisors, or the VMs etc. However, for simplicity, we consider that the OpenFlow devices are equipped with IDS agents. The IDS has updated signatures. It hosts a Reporting Agent. The purpose of this agent is to inform various event to the Report Aggregator module.

G. C5: Report Aggregator

The Report Aggregator is an event logger. It analyses the event logs to find the causalities between the events. It is a time-

oriented policy-based event analysis module. This module can be made more intelligent with a machine learning algorithm. We plan to extend this work in the journal version of the paper.
Event Detection Mechanism: This module holds policies associated with different events. For instance, event 1 is a case where a user spoofs his address and event 2 is a case where he scans the network. If both cases occur from the same host or IP, the Report Aggregator raises an alert.
These malicious and suspicious events are reported back to the APbSA. The APbSA takes action accordingly.

H. C6: Key Management Mechanism

This module is responsible for maintaining secure communication between the Controller and the OpenFlow switches. It also helps in the generation and distribution of keys for secure flow communication. The secure flow communication means securing the payload as well as providing end-to-end authentication for flow communication. Thus, it allows confidentiality and integrity services to user flows. The methods used in this module are standard encryption mechanisms and is not a special contribution to the work. Now, we will provide a brief description of the task performed by this module.

For Secure Flow Installation: Although OpenFlow has a secure version [13], we focus on using a different TLS based approach for secure flow installation. We have used public-key cryptography for this purpose. At first, the Controller and the OpenFlow switches negotiate on the cipher suite for communication. This happens during the switch boot time and is implemented over TLS. Once they have agreed on a suite, cryptographic keys are generated. Finally, the Controller uses these cryptographic keys for secure flow installation.

For Securing Flow Information: APbSA provides user data confidentiality services. As part of it, the Key Management Module can encrypt the flow payloads while transmitting them. It uses Symmetric Key Cryptography. This is an on-demand network service and is associated with specific policies in the APbSA. For instance, let us assume that user A has requested a confidentiality service and the endpoints of his communication are $SW1$ (source switch) and $SW4$ (destination switch). APbSA, based on the policy, will generate a symmetric key and will distribute it to $SW1$ and $SW2$. The switch agents will use this key to encrypt the payload at the source switch and decrypt the payload at the destination switch.

For End-to-end Authentication: The purpose of this mechanism is to check the authenticity of the flow sender and stop the propagation of fake flows. It uses a public key encryption protocol. Using this approach, every intermediary switch in a routing path can justify which OpenFlow switch is sending a particular flow. This improves the authenticity of the routing path and OpenFlow switches can judge forged routing request and flows. As the scope is limited to explain this work, we plan to explain this protocol in the journal version of the paper.

I. Dynamic Policy Generation:

APbSA can generate policies dynamically, which helps in creating a secure network infrastructure and secure routing environment. In this section, we will present how APbSA generates policies.

$$PE^{D_i} = < FlowID, SourceAS, DestAS, SourceHostIP, DestHostIP, SourceMAC, DestMAC, User, FlowCons, DomCons, Services, Sec - Profile, Path >: < drop/reroute/reset > \quad (1)$$

The Report Aggregator reports suspicious events to the APbSA. Then APbSA determines the identities of the devices associated with this event. It can be a user or a device containing IP, MAC, service and ID. After that, APbSA will construct a rule; in this case, a drop rule with all the malicious event creator's identity. The generic syntax for the dynamic policy is shown in equation (1). APbSA enforces these policies depending on the origination point. There can be two types of origination point:

- If the originator is an external user or an IoT device, the drop rule will be installed in the nearest connecting OpenFlow switch. This will stop the propagation of the attack at the origination point. For instance, user A, with IP address $172.168.61.10$ is generating suspicious traffic; then the drop rule is:
  $PE^{D_{10}} = < *, *, *, 172.168.61.10, *, *, *, A, *, *, *, *, * >: < drop > \quad (2)$
- If the suspicious entity is a switch surrounded by neighbouring switches, APbSA will isolate the infected switch. APbSA does that by rewriting new flow rules in the neighbouring switches.

III. Implementation

In this section, we will first present our network setup for the proof of concept prototype, network performances and security analysis with scenarios.

A. Network Setup

We have used a VMware NSX cluster of the Advanced Cyber Security Engineering Research Centre (ACSRC) at the University of Newcastle to form the network. The cluster uses three Dell Power Edge M640 Server, consisting of two Intel Xeon Gold 6126 @ 2.6 GHz processors and 384 GB(6x64) of RAM. We have used ONOS as the SDN Controller. The APbSA and other network applications are developed as a part of ONOS Controller and are assumed trusted. The Mininet Virtual Machine creates the switching network. For capturing network activity and logs, we have used the Security Onion Virtual Machine and Splunk [14] in this POC network setup. Splunk is a real-time data/log management solution [14].

B. Network Performance:

We have measured two types of performances for this prototype. As the OpenFlow switches are modified to perform some additional activities, we will present heap memory usage by a single OpenFlow switch. For the overall network, we will provide the network throughput with our ISA architecture.

1) OpenFlow Switch Memory Overhead

We have used Massif [15] to gather the memory used by different processes within the OpenFlow switch. It is a command-line based memory profiler for Linux processes. The ISA uses a modified OpenFlow switch with custom agents for traffic validation and event reporting which consumes extra memory.

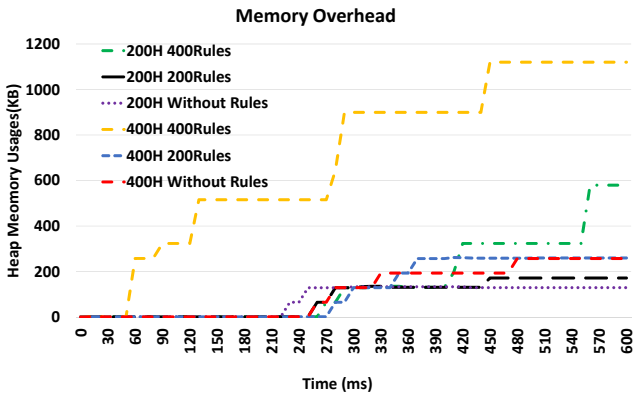


Fig. 3: Memory Overhead

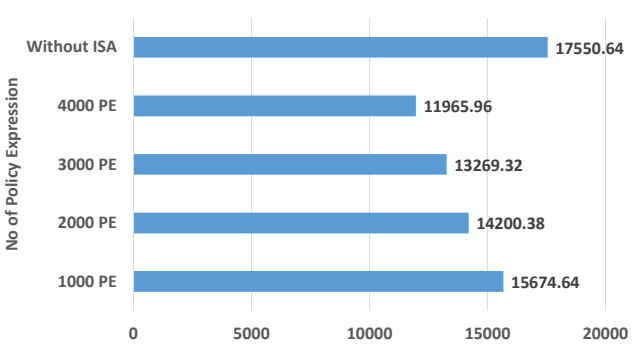


Fig. 4: Average Throughput

Hence, the memory usage varies with the end hosts connected to a switch and with their flows. Figure 3 presents the memory usage with a varying number of end hosts connected to the switches. In all the cases, we have noticed a gradual increase in memory usage, which becomes stable after some time. However, the memory usage increases with the increase in the number of end hosts and the number of rules for validating the traffic.

2) Network Throughput

To measure the throughput, we have used cBench. We have created a network with eight OpenFlow switches and each of them are connected with 200 dummy hosts. Figure 4 shows the throughput comparison while using ISA and in normal SDN enterprise network. From the overall perspective, it is evident that the throughput decreases with the increase in the Policy Expression.

C. Security Analysis

Here, we will provide example attack scenarios to explain the effectiveness of ISA and present how it fulfils the security requirements mentioned in II-A.

1) Attack Scenario

Figure 5 shows a healthcare network infrastructure with wireless medical devices. These devices can track changes related to a patient's health. The network infrastructure consists of various servers hosted over a virtualized server. It has the following: i) a web server to provide an interface to the health record; ii) a database server for storing patient records; iii) an application server for hosting web applications; and iv) a

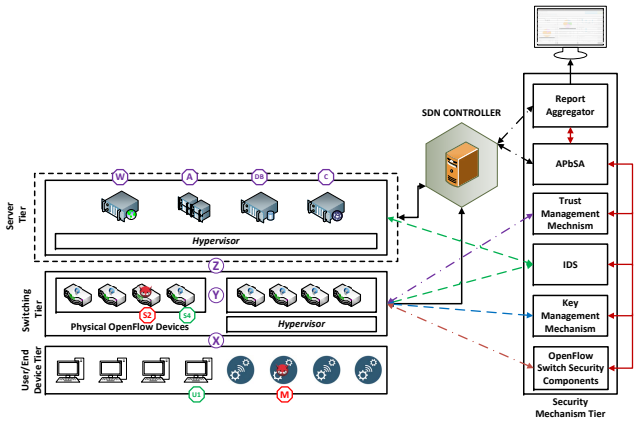
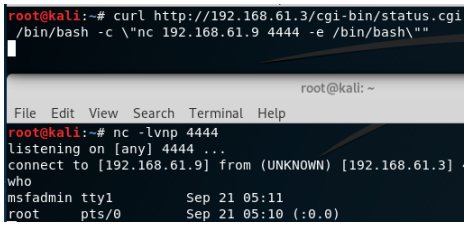
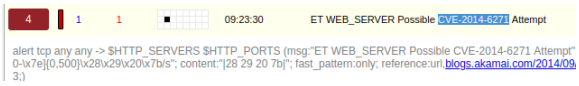


Fig. 5: A healthcare network infrastructure

content management server for storing and managing patient profiles. The patient records are valuable and the network should be highly secure. In a normal SDN enabled healthcare network, a patient monitoring device (M) and user ($U1$) can request for a flow route towards the database server and web server, respectively, to the OpenFlow switching devices. If the flow rules are not present in the OpenFlow devices, it will request the Controller via *packet_IN* request and after installing the flow rules they can establish communication. Let us assume that it is a normal SDN network and (M)



(a)



(b)

Fig. 6: a) Successful attack and b) Alert in ISA

is a malicious patient monitoring device. Every device is authenticated while they are connected. However, M was physically tampered and uses an old Linux with minimal security. The target of the adversary was to deploy malware that injects a vulnerable CGI file in an OpenFlow switch, $S2 : IP - 192.168.61.3$, which will allow privilege escalation attacks like shellshock. We assume that the malware was successfully deployed. The adversary was successful in getting the backdoor (Figure 6a). With that privilege, he was able to inject fake flow rules in the OpenFlow switches and reroute the packets to a different IP address. Figure 6a shows a successful shellshock attack. In contrast, with ISA, the attack was detected by the agents and alert was raised (as shown in Figure 6b). Also, APbSA will detect unauthorised communication and block the communication with a new flow dropping rule. Furthermore, the trust level of the infected switch will go down. The report aggregator will observe these events. Finally,

APbSA will isolate the infected switch by rewriting new flow rules in the neighbouring switches.

2) Discussion

The architecture satisfies the following requirements:

- **R1:** ISA is an interconnected, feedback architecture, which takes into account the various agent reports in formulating the policy. Hence it can achieve the coordination of different types of security mechanisms in an enterprise network.
- **R2:** The agent reports from different layers and Report Aggregator interconnects the suspicious network events. This approach is helpful in the detection of zero-day vulnerabilities. Also, it can detect flaws associated with different paths and routing for dynamic policy-based patching.
- **R3:** ISA uses a subjective logic-based trust evaluation mechanism for evaluating the trustworthiness of the switches. This trust model enables the Controller to determine the trust levels associated with the swithes, which are then used in the formulation of security policies to specify the flows over the network.
- **R4:** ISA with the help of Key Management Module provides confidentiality and integrity services required to protect the flows.
- **R5:** ISA is able to detect a range of security attacks such as DDoS, spoofing, attacks specific to SDN switches etc.

IV. Related Works

A. Policy and trust for SDN Security

PANE [16] is a prototype SDN Controller which focuses on end-user requests for providing various network services, such as performance, security etc. The whole process is dynamic and they use policies to capture the end-user requests and enforce them using PANE SDN Controller. ISA does not consider any direct input from the end-user. However, network administrators can convert user preferences into flow and path-based policies.

Flow-based Management Language (FML) [17], Frenetic [18], Pyreytic [19], Maple [20] and Nettle [21] are the most prominent SDN policy languages. They are not dynamic and complex for any human administrator to write. Moreover, these languages do not focus on the security aspects of the SDN enterprise network, rather on issues related to network performances and routing. On the other hand, ISA mainly focuses on providing dynamic security to the SDN enterprise network. It uses security policies, trust and attacks detection mechanisms to make the enterprise network safe.

Li et al. in [22] have presented a dynamic access control policy enforcement mechanism for SDN. It uses anomaly-based IDS to detect attacks in SDN. Then the attacker's attributes are used to form dynamic access control policy. The SDN Controller enforces these policies. Our work extends the idea of dynamic policy enforcement for SDN. Our ISA deploys different types of mechanisms to assess the status of various components in the SDN enterprise network. This involves both attack detection and false routing for devices. Thus, helping in the generation of fine granular routing control and safeguarding policies for the SDN infrastructure.

Manso et al. in [23] presented an SDN IDS system for early detection of DDoS attacks. Their approach uses traffic rate as the detection metrics and once detected, they have programmed the Controller to block malicious flows by installing new flow rules. Our approach has switch agents who report the status of the switches, thus enabling APbSA to counteract the attacks by installing dynamic policies.

Very few works focus on the trust issues of the SDN infrastructure. One of them is [24] where the authors have used the Controller to evaluate the trustworthiness of the network applications. The authors have introduced a theoretical trust establishment framework for network applications. Here, the Controller maintains a table containing permissions for different Controller resources for separate network applications. Based on the network applications previous requests for various resources, the Controller evaluates the direct trust for any network application. A trust negotiation protocol between different trusted domain are presented in [25]. Here, authors have focused on authentication based trust establishments between the domains. In contrast, our security architecture focuses on the trustworthiness of the network devices such as the gateways, switches and routers etc. Also, due to the integrated dynamic policy mechanism, it can take dynamic actions to isolate the untrusted switch.

B. Attacks in SDN

SDN enterprise networks are vulnerable to security issues [5, 26, 7]. In these research works, the authors have mentioned various types of vulnerabilities at three layers of the SDN architecture. Our work provides a combined solution to tackle various security issues in SDN enterprise network.

V. Conclusion

In this paper, we have presented an Integrated Security Architecture for SDN based enterprise networks. A distinguishing characteristic of the proposed architecture is its ability to deal with dynamic changes such as changes in the security attacks or the trust associated with the various components or services in the infrastructure. This leads to a security architecture that can adapt to dynamic changes, which is achieved by having feedback between the various security components/mechanisms in the architecture and managed by a dynamic policy framework. Our architecture uses dynamic policy based approach at the SDN Controller and enforces security mechanisms in the switches to counteract a range of security attacks in the SDN infrastructure. We have described the prototype of our architecture using a VMware NSX cluster, and presented security and performance analysis for different attack scenarios. We believe that the proposed integrated security architecture provides a significant step towards achieving a secure SDN for enterprises.

Acknowledgement The authors would like to thank Defence Science and Technology (DST) Group for their financial contribution to this work under the NGTF Cyber Program. In particular, we would like to thank Dr Peter Dickinson and his Group. The DST funding should not be taken to imply endorsement of the contents or conclusions of the paper.

REFERENCES

[1] D. Levin *et al.*, "Incremental sdn deployment in enterprise networks," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 473–474.

[2] M. Casado *et al.*, "Sane: A protection architecture for enterprise networks." in *USENIX Security Symposium*, vol. 49, 2006, p. 50.

[3] R. P. Lippmann *et al.*, "Evaluating and strengthening enterprise network security using attack graphs," *Lexington, Massachusetts October*, 2005.

[4] "Cybersecurity management and oversight at the jet propulsion laboratory," https://oig.nasa.gov/docs/IG-19-022.pdf.

[5] K. Benton *et al.*, "Openflow vulnerability assessment," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in SDN*. ACM, 2013, pp. 151–152.

[6] D. Kreutz et al., "Towards secure and dependable software-defined networks," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13. ACM, 2013, pp. 55–60.

[7] S. Scott-Hayward *et al.*, "Sdn security: A survey," in *2013 IEEE SDN For Future Networks and Services*. IEEE, 2013, pp. 1–7.

[8] K. K. Karmakar *et al.*, "Sdn enabled secure iot architecture," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 581–585.

[9] B. WU JP *et al.*, "Rfc 5210, a source address validation architecture (sava) testbed and deployment experience," 2008.

[10] A. Jøsang, *Subjective Logic: A Formalism for Reasoning Under Uncertainty*, 1st ed. Springer, 2016.

[11] A. Jøsang *et al.*, "Legal reasoning with subjective logic," *Artificial Intelligence and Law*, vol. 8, no. 4, pp. 289–315, 2000.

[12] K. Karmakar, "Techniques for securing software defined networks and survices," Ph.D. dissertation, University of Newcastle, Australia, 2019.

[13] A. Azzouni *et al.*, "softdp: Secure and efficient openflow topology discovery protocol," in *IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–7.

[14] J. Stearley *et al.*, "Bridging the gaps: Joining information sources with splunk." in *SLAML*, 2010.

[15] "massif, memory profiler," https://courses.cs.washington.edu/courses/cse326/05wi/valgrind-doc/ms_main.html.

[16] A. D. Ferguson *et al.*, "Participatory networking," in *Presented as part of the 2nd {USENIX} Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, 2012.

[17] T. L. Hinrichs et al., "Practical declarative network management," in *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 2009, pp. 1–10.

[18] N. Foster et al., "Frenetic: A network programming language," in *ACM SIGPLAN Notices*, vol. 46, no. 9. ACM, 2011.

[19] J. Reich et al., "Modular sdn programming with pyretic," *Technical Reprot of USENIX*, 2013.

[20] A. Voellmy et al., "Maple: simplifying sdn programming using algorithmic policies," in *ACM SIGCOMM Computer Comm. Review*, vol. 43, no. 4. ACM, 2013, pp. 87–98.

[21] ——, "Nettle: Taking the sting out of programming network routers," in *Practical Aspects of Declarative Languages*. Springer, 2011, pp. 235–249.

[22] H. Li *et al.*, "Enabling dynamic network access control with anomaly-based ids and sdn," in *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. ACM, 2019.

[23] P. Manso *et al.*, "Sdn-based intrusion detection system for early detection and mitigation of ddos attacks," *Information*, vol. 10, no. 3, p. 106, 2019.

[24] B. Isong *et al.*, "Trust establishment framework between sdn controller and applications," in *18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*. IEEE, 2017.

[25] R. Zhou *et al.*, "Study on authentication protocol of sdn trusted domain," in *2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems*. IEEE, 2015, pp. 281–284.

[26] D. Kreutz *et al.*, "Sdn: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.